UNITED STATES PATENT AND TRADEMARK OFFICE

MN

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/087,296	03/01/2002	Anil Seth	1488.008US1	2127

21186 7590 07/12/2007
SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. BOX 2938
MINNEAPOLIS, MN 55402

EXAMINER	
ROMANO, JOHN J	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
07/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

MAILED

JUL 12 2007

Technology Center 2100

Application Number: 10/087,296
Filing Date: March 01, 2002
Appellant(s): SETH ET AL.

David D'Zurilla
Reg. No. 36,776
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed March 9th, 2007 appealing from the Office action mailed January 3rd, 2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,219,796	Bartley	04-2001

Y. Li and J. Henkel. A framework for estimating and minimizing energy dissipation of embedded hw/sw systems. In Proceedings of the Design Automation Conference, pages 188-193, 1998.

G. Ramalingam. Data flow frequency analysis. In Proceedings of the SIGPLAN '96 conference on Programming Language Design and Implementation, pages 267-277, May 1996.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

- Claims **1, 2, 11-15, 22-25, 32-36, 43** and **44** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bartley, US 6,219,796 (hereinafter **Bartley**), in view of Y. Li et al. A framework for estimating and minimizing energy dissipation of embedded hw/sw systems, (hereinafter **Li**).

In regard to claim **1**, **Bartley** discloses:

- "*A method of compiling computer code including power-down instructions to reduce power consumption during execution of the*

code..." (E.g., see Figure 7 & Column 2, lines 62-67), wherein it is inherent that the code is efficient when executed by a processor.

- "*...identifying one or more potential locations in the computer code where the power-down instructions can be inserted...*" (E.g., see Figure 7 & Column 7, lines 10-21), wherein the potential locations are identified by scanning the code.
- "*...selecting locations to insert the power-down instructions from the identified potential locations in the code based on reducing power consumption ...*" (E.g., see Figure 7 & Column 7, lines 39-43), wherein the locations are determined by a predetermined threshold duration of non-use.
- "*...inserting the power-down instructions in the selected locations to reduce the power consumption during the execution of the code ...*" (E.g., see Figure 7 & Column 7, lines 43-46), wherein the power modifying or power-down instruction is then inserted to reduce the power consumption.

But **Bartley** does not expressly disclose "*...satisfying user-specified real-time constraints...*". However, **Li** discloses:

- "*...satisfying user-specified real-time <u>performance</u> constraints...*" (E.g., see Figure 5 & Page 4, Section 4.3), wherein the user specifies one of many multiple objective optimization goals via performance constraints.

Bartley and **Li** are analogous art because they are both concerned with the same field of endeavor, namely, an optimizing compiler with the means to reduce power or energy consumption. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine user specified real-time constraints with **Bartleys'** power reduction methods. The motivation is disclosed by **Bartley,** as he refers to program segments having a duration longer than a "predetermined threshold." (Column 7, lines 42-43), wherein it is obvious the threshold may be determined by a user either via a user selected algorithm or other user input.

In regard to claim **2**, the rejections of base claim **1** are incorporated. Furthermore, **Bartley** discloses:

- "...*wherein the code is written for a microprocessor having distinct functional units*." (E.g. see Figure 7 & Column 3, lines 3-8) wherein the common characteristic is any processor or microprocessor that has more than one independent or distinct functional units.

In regard to claim **11**, the rejections of base claim **1** are incorporated.

Furhtermore, *Li's* teachings of "minimum energy dissipation while not exceeding <u>the budget of clock cycles</u>" (execution time constraint), *See Li*, Section "5.2 Optimizing system-level energy dissipation"; is relied upon to have been obvious to one of ordinary skill in the art to teach the "number of additional cycles of execution time the user is willing to incur" due to a software transformation.

The claim limitations of "number of power down instructions that can be inserted in an execution path" is determined by *Bartley* as disclosed in claim 1 (See previous

rejection or section "iv") above), wherein *Bartley's* disclosure of determining potential locations to insert power down instructions along with subsequently determining where to insert the instructions from the identified potential locations, inherently or necessarily determine the "number of power down instructions that can be inserted in an execution path, including one or more identified potential locations".

Furthermore, **Li** discloses:

- "*... the number of ...instructions that can be inserted in an execution path, including one or more identified potential locations.*" (E.g. see Table 2 & Section 5.2), wherein the time improvement or a negative time improvement as a performance constraint is taught and may be used to limit the number of instructions inserted.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine **Li's** user specified real-time constraints with **Bartleys'** power reduction methods. The motivation is disclosed by **Bartley,** as he refers to program segments having a duration longer than a "predetermined threshold." (Column 7, lines 42-43), wherein it is obvious the threshold may be determined by a user either via a user selected algorithm or other user input. Furthermore, the segment is a direct relationship to **Li's** teaching of user specified performance constraint of time or execution cycles executed as a consequence of the energy savings. Additionally, **Bartley** provided the motivation for a number of power down instructions (E.g. see, Figure 5 & Column 2, line 11) wherein, it would have been obvious to one of ordinary

skill in the art, to factor in particular power down instructions and the number of such instructions, based on the energy savings in relation to the overhead drawback.

In regard to claim **12**, the rejections of base claim **11** are incorporated. Furthermore, **Li** discloses.

- "*...the number of additional cycles of execution time the user is willing to incur...*" (E.g. see Table 2 & Section 5.2), wherein the "...minimum energy dissipation while not exceeding the budget of clock cycles to execute..." is taught.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine user specified real-time constraints with **Bartleys'** power reduction methods. The motivation is disclosed by **Bartley,** as he refers to program segments having a duration longer than a "predetermined threshold." (Column 7, lines 42-43), wherein it is obvious the threshold may be determined by a user either via a user selected algorithm or other user input.

In regard to claim **13**, the rejections of base claim **11** and claim **12** are incorporated. Furthermore **Bartley** discloses:

- "*...inserting power-up instruction in the code to restore at least one functional unit to a ready state powered-down by the inserted power-down instructions..*" (E.g. see Figure 7 & Column 6, lines 8-19), wherein the power up instruction is inserted.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine **Li's** user specified real-time constraints

with **Bartleys'** power reduction methods. The motivation is disclosed by **Bartley,** as he refers to program segments having a duration longer than a "predetermined threshold." (Column 7, lines 42-43), wherein it is obvious the threshold may be determined by a user either via a user selected algorithm or other user input. Additionally, the segment is a direct relationship to **Li's** teaching of user specified performance constraint of time or execution cycles executed as a consequence of the energy savings.

As per claims **14, 15, 22** and **23,** this is a computer-readable medium version of the claimed method discussed above, in claims **1, 2, 11** and **13,** wherein all claimed limitations have also been addressed and/or cited as set forth above, wherein **Bartley** also discloses "a storage device and external memory" (16), (E.g. see, Figure 1 and associated text).

As per claims **24, 25, 32** and **33,** this is a computer system version of the claimed method discussed above, in claims **1, 2, 11** and **13,** wherein all claimed limitations have also been addressed and/or cited as set forth above, wherein **Bartley** also discloses a computer system (E.g. see, Figure 1 and associated text).

In regard to claim **34**, the rejections of claim **1** are incorporated. Additionally, **Bartley** discloses:

- "*A computer readable medium having a computer program including instructions for causing a computer to perform a method of selectively controlling power to different functional units of the computer, the instructions comprising...*" (E.g., see Figure 7 & Column 7, lines 10-

21), wherein it is inherent that the instructions have to be on a computer-readable medium to be scanned by a computer process.

- "*...power-down instructions inserted in the computer-program in selected locations based on reducing power consumption...*" (E.g., see Figure 7 & Column 7, lines 10-21), wherein the potential locations are identified by scanning the code.
- "*...the power-down instructions in the selected locations reduce the power consumption during the execution of the code...*" (E.g., see Figure 7 & Column 2, lines 6-13), wherein the locations are determined by a predetermined threshold duration of non-use.

As per claims **35, 36, 43** and **44,** the base claim **34** is incorporated. Furthermore, this is another computer-readable medium version of the claimed method discussed above, in claims **1, 2, 11** and **13,** wherein all claimed limitations have also been addressed and/or cited as set forth above, (E.g. see Figure 1 & associated text), wherein a computer readable medium is shown (16).

- Claims **3-10, 16-21, 26-31** and **37-42** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bartley** in view of **Li** and further in view of G. Ramalingam. Data Flow Frequency Analysis, SIGPLAN Conference on Programming Language Design and Implementation, 1996, (hereinafter **Ramalingam**).

In regard to claim **3**, the rejections of base claim **2** are incorporated. Furthermore, **Bartley** discloses:

- "*... based on the functional units not being used in the potential locations, wherein the functional units not being used are determined based on functional unit usage ...*" (E.g. see Figure 7 & Column 7, lines 10-21), wherein the functional units are not used.

But **Bartley** does not specifically disclose a "*...transfer functions at each of the potential locations as specified in standard monotone data-flow frameworks.*" However, **Ramalingam** discloses:

- "*...transfer functions at each of the potential locations as specified in standard monotone data-flow frameworks.*" (E.g. see Section 3, The expected Frequency of Dataflow Facts), wherein the use of transfer functions as specified in standard monotone data-flow frameworks is taught.

The combined teaching and **Ramalingam** are analogous art because they are both concerned with the same field of endeavor, namely program optimization via standard analysis. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine a transfer function with static analysis method disclosed by the combined art of an optimizing compiler embodiment. The motivation is disclosed by **Bartley,** "Locating program segments during which a functional unit is not used may be done by either static or dynamic program analysis." (Column 7, lines 47-49).

In regard to claim **4**, the rejections of base claim **3** are incorporated. Furthermore, **Bartley** discloses:

- "... *statically analyzing processor cycles prior to executing the code.*" (E.g. see Figure 7 & Column 7, lines 47-52), wherein the processor or execute cycles are estimated by the compiler for static analysis.

In regard to claim **5**, the rejections of base claim **4** are incorporated. Furthermore, **Bartley** discloses:

- "...*the text in the code...*" (E.g. see Figure 7 & Column 7, lines 47-52), wherein the start and stop points exist in the program segments or text in the code.

In regard to claim **6**, the rejections of base claim **3** are incorporated. Furthermore, **Bartley** discloses:

- "...*a first power-down instruction operable to reduce power to all of the at least one functional unit, such that the functional unit is placed in a low state of readiness and a second power-down instruction operable to reduce power to only a part of the at least one functional unit, such that the functional unit is placed in an intermediate state of readiness.*" (E.g. see Figure 6 & Column 6, line 60 – Column 7, line 3), wherein the "less ready" or low state and a "more ready" or intermediated state of readiness are taught.

In regard to claim **7**, the rejections of base claim **1** are incorporated. But Bartley does not expressly disclose "...*executing the code to generate power-profiling and execution path-profiling information...*" or "...*assigning a weight factor based on the profile information...*". However, **Li** discloses:

- "*...executing the code to generate power-profiling information associated with each of the identified potential locations...*" (E.g. see Figure 2 & Page 3, Section 3.4), wherein Figure 2 shows the program execution trace which generates the software performance model and the software energy model is also generated based on the execution trace and then coupled with the memory energy models to account for the total system energy generating power information or a power-profile.
- "*...assigning a weight factor to each of the identified potential locations based on the generated power-profiling...*" (E.g. see Figure 5 & Section 4.2), wherein the EES/CSI ratio or weight factor prioritizes and then gets assigned a probability based on the ratio. Further the EES/CSI numbers are based on the profile information. Additionally, the user specifies constraints to be met in real-time in section 4.3.

But the combined teaching of **Bartley** and **Li** do not expressly disclose "...executing the code to generate path-profiling information...". However, **Ramalingam** discloses:

- "*...path-profiling information...*" (E.g. see Section 1), wherein the path-profiling information is used to estimate probability.
- "*...and path-profiling information; and selecting the locations to insert the power-down instruction from the identified locations based on the*

assigned weight factors..." (E.g. see Section 3, lemma 2), wherein the result is "...weighted...".

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine power and path profile information with **Bartleys'** power reduction methods. Motivation was provided by **Bartley,** when he referred to static and dynamic analysis utilizing execution cycles, loop cycles and other "statistical predictions." (Column 7, lines 47-52), wherein it would have been obvious, at the time the invention was made, that **Li's** constraints and profile algorithm would be beneficial to the efficiency of a power reduction embodiment disclosed by **Bartley**. Furthermore, motivation was provided by **Li** (Figure 2) wherein, the program execution trace used by **Li** would only been beneficial if there was a probability that the path will actually be used.

In regard to claim **8**, the rejections of base claim **7** are incorporated. Furthermore, **Li** discloses:

- "...*generating execution probability of each of the identified potential locations based on the generated path-profiling information.*" (E.g. see Section 3, lemma 2), wherein the result is "...weighted..." by the probability of execution of the path.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine probability derived from path profile information with **Bartleys'** power reduction methods in order to increase the efficiency by increasing the depth of the analysis.

In regard to claim **9**, the rejections of base claim **8** are incorporated. Furthermore, **Li** discloses:

- "*...extracting potential energy savings for each of the identified potential locations using the generated power profile analysis information...*" (E.g. see Figure 5 & Page 4, Section 4.2), wherein the EES is the estimated energy savings.
- "*...assigning the weight factor to each of the identified potential locations based on the extracted potential energy savings and the generated execution probability.*" (E.g. see Figure 5 & Page 4, Section 4.2), wherein the EES/CSI ratio or weight factor prioritizes and then gets assigned a probability based on the ratio. Further the EES/CSI numbers are based on the program execution trace or generated path-profiling information.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine potential energy savings derived from power profile information with **Bartleys'** power reduction methods in order to increase the efficiency by increasing the depth of the analysis.

In regard to claim **10**, the rejections of base claim **9** are incorporated. Furthermore, **Li** discloses:

- "*...executing the code to assign a first weight factor based on the extracted potential energy savings to each of the identified potential locations...*" (E.g. see Figure 2 & Column 3, lines 3-8), wherein the

software performance model includes the product of execution cycles of a given instruction and the number of times an instruction is used or path profile and power information is factored to derive a weight factor.

- "*... executing the code to assign a second weight factor based on execution probability at each of the identified potential locations...*" (E.g. see Figure 2 & Column 3, lines 3-8), wherein the software performance model includes the product of execution cycles of a given instruction and the number of times an instruction is used or path profile.
- "*...computing a product of the first and second weight factors for each of the identified potential locations; calculating the weight factor for each of the identified potential locations based on the computed product of the first and second weight factors; and assigning the calculated weight factor to each of the identified potential locations.*" (E.g. see Figure 2 & Column 3, lines 3-8), wherein the software performance model includes the product of execution cycles of a given instruction and the number of times an instruction is used or path profile and the weight factor is assigned based on a product of weighted factors of both the energy savings or power profile and execution probability. The EES/CSI ratio as disclosed above is based on the products of path and profile information.

As per claims **16-21,** this is a computer-readable medium version of the claimed method discussed above, in claims **3, 4** and **7-10,** wherein all claimed limitations have also been addressed and/or cited as set forth above, (E.g. see Figure 1 & associated text), wherein a computer readable medium is shown (16).

As per claims **26-31,** this is a computer system version of the claimed method discussed above, in claims **3, 4** and **7-10,** wherein all claimed limitations have also been addressed and/or cited as set forth above, (E.g. see Figure 1 & Column 3, lines 3-8), wherein a computer system is shown.

As per claims **37-42,** the base claim **34** and **35** are incorporated. Furthermore, this is another computer system version of the claimed method discussed above, in claims **3, 4** and **7-10,** wherein all claimed limitations have also been addressed and/or cited as set forth above, (E.g. see Figure 2 & Column 3, lines 3-8), wherein a computer system is shown.

(10) Response to Argument

A) *Examiner's response to Appellant's discussion of Claims 1, 2, 11-15, 22-25, 32-36, 43 and 44 (See brief, pages 9 – 13).*

i) Appellant argues (*See* brief, page 11, 1st, paragraph):

"No mention is made of code performance in Bartley, only efficient use of power."

In response to applicant's argument that the *Bartley* fails to show certain features of applicant's invention (i.e., "code performance"); it is noted that *Bartley* is not relied upon by examiner. Rather, *Li* is relied upon (*See* final rejection, mailed 1/03/07, page 11) for said instant limitation. As relied upon in the previous rejection cited above, and reproduced herein, *Li* discloses:

- "*...satisfying user-specified real-time performance constraints...*" (E.g., see Figure 5 & Page 4, Section 4.3), wherein the user specifies one of many multiple objective optimization goals via performance constraints.

ii) Appellant argues (See brief, page 11, 3rd paragraph) :

"Applicant does not believe that the references are properly combinable, as each is directed to very different aspects of power reduction."

While *Li's* disclosure does include modifying hardware as argued by Appellant (*See* brief, page 11, 2nd paragraph), it is noted that this is in the context of exploring the design space or modifications of the hardware as a result of software transformations (i.e., "the trade-off in energy dissipation among software [1], memory and hardware", wherein the term energy dissipation is defined as the energy dissipated within a processor core (*See Li*, "Introduction"). This teaching does not mean that a particular software transformation / optimization may not be considered. In fact, *Li* expressly discloses "To optimize the system energy, we explore the design space in the dimensions of software and cache/memory. As mentioned in Sec.3, our framework

assumes that the hardware (ASIC) is fixed. It changes the software by performing various high-level transformations." (emphasis added - See Li, page 3, "4 System-level Energy Optimization"). Herein it seems clear that software is transformed or modified/ changed.

Additionally, *Li* expressly discloses "The algorithm is independent of the transformations themselves" (emphasis added - *See* page 4, second paragraph). At this point, it should be clear that software is changed /transformed to optimize energy. Accordingly, combining *Li's* disclosure with a software transformation / modification to optimize energy would certainly be of interest to one of ordinary skill in the art. As a result, including a software optimization comprising power down instructions as disclosed by *Bartley* would certainly be inline with the modification /transformation of software to optimize energy.

The fact that *Li* then discloses evaluating the run-time performance of the system in response to those modifications/ changes only further shows the similarity to Appellant's invention rather than teaching away from it. This is evident by Appellant's own description of claim 1, (*See* brief, page 10, third paragraph), wherein Appellant states "As indicated above in claim 1, the present claims allow a tradeoff between performance and power conservation based on user specified constraints for execution of program code". Similarly, as addressed herein-above in more detail "The trade-off between system performance [execution performance] and energy dissipation [power conservation] is also explored." (*See Li*, Abstract). Thus, the combination of *Li's*

teaching with *Bartley's* software transformation directed towards energy consumption is indeed proper.

iii) Appellant argues (See brief, page 12, 1st paragraph):

> ""Various power modeling techniques can be used to determine the length of time during which it is more efficient to turn a component off (or partially off) then on again versus leaving it on." Col. 7, lines 16-19. It does not relate directly to satisfying user-specified real time constraints or program performance as currently claimed. As such, it would not suggest to one of ordinary skill in the art that performance optimization goals should be considered."

The plain language of the claims merely recite "satisfying user-specified real-time performance constraints". It is noted that there is no express or deliberate definition of "user specified real-time performance constraint" in Appellant's specification. Accordingly, with respect to claim 1, the examiner interprets a "user-specified performance constraint", in light of the plain meaning of the claim limitations, as a run-time performance measure that needs to be met and that is specified by the user. As such, *Li's* teaching under section system-level energy optimization algorithm, wherein a <u>user specifying</u> one optimization goal (emphasis added - *See Li*, section 4.3, item "4"), wherein the user may specify "Goal II: minimized power under <u>performance constraints</u>" (emphasis added - *See Li*, 4.3, Goal II). It is noted that the minimization of power is defined as the software energy consumption by the processor under certain system parameters (See "introduction", footnote 1), but particularly <u>performance constraints</u> /parameters. The user also may choose multiple objective <u>optimizations</u> as expressly

taught in Goal III, wherein, a set of solutions within performance and energy constraints (*See Li*, 4.3, Goal III), is specified by the user. The user / designer then chooses the most suitable solution that have met the specified performance constraints. Thus, the user choosing an energy optimization algorithm, which comprises a specified performance constraint certainly reads on the plain language of the instant claim limitation ("user specified real-time performance constraint").

Even arguably, as defined in the originally filed disclosure, exemplary of such user specified real-time constraints simply states "The user-specified real-time constraints can include constraints such as the number of power down instructions that can be inserted in an execution path, the number of additional cycles of execution time the user is willing to incur, and other such constraints" (emphasis added - *See* specification, page 12, lines 10-13).

That is to say that the aforementioned code *performance constraints* as herein argued by Appellant can include the program reducing power by constraining execution instructions, possibly by the user specifying additional cycles of execution to determine execution time and other such constraints. Appellant defines an embodiment of a user-specified real-time constraints comprising the user specifying a number of execution cycles, to compute the threshold of execution time used to insert a power down instruction, expressly referred to by the specification as "***execution time constraint***" (See specification, page 12, lines 20-23, italicized emphasis in original specification – bolded and underlined emphasis by examiner).

Accordingly, "performance constraints", even arguably, in its disclosed detailed embodiment, wherein Appellant equates the user specifying clock cycles and refers to the allotment of clock cycles as "execution time constraint" as addressed above, is suggested by **Li's** teachings of "minimum energy dissipation while not exceeding the budget of clock cycles" (execution time constraint), *See Li*, Section "5.2 Optimizing system-level energy dissipation"; wherein, *Li* further supports the above interpretation of the examiner. Although *Li*, does not expressly state that the user specifies the budget of clock cycles, the user / programmer specifies the goal which includes the budget of clock cycles and therefore specifies not exceeding the budget /allotment of clock cycles to execute as expressly disclosed by *Li* in section 5.2, and illustrated in Table 2. Also, in order to have a "budget" the programmer /user inherently must specify the budget which again should be noted, is equated by Appellant to equal "execution time constraint".

Therefore, it seemed appropriate to the examiner to use *Bartley's* teaching of choosing one of several known algorithms to compute the execution time threshold for inserting power down instructions as a suggestion to combine *Li's* teaching of performance constraints (energy optimization in view of design constraints).

iv) Next, Appellant concludes the above argument by stating (See brief, page 12, 1st paragraph):

"In practice, with the presently claimed invention, there may be many places in code where a power down instruction could be added. The claimed invention allows one to determine where to put them to optimize power consumption within user specified constraints."

It is unclear to the examiner, what Appellant means by the above statement or how it even relates to Appellant's preceding argument, quoted by examiner above. However, in regard to determining where to insert power down instructions, see the previous rejection, wherein *Bartley* discloses determining where to put power down instructions to optimize power consumption. Reproduced herein-below for convenience.

In regard to claim **1**, **Bartley** discloses:

- "*A method of compiling computer code including power-down instructions to reduce power consumption during execution of the code*..." (E.g., see Figure 7 & Column 2, lines 62-67), wherein it is inherent that the code is efficient when executed by a processor.
- "...*identifying one or more potential locations in the computer code where the power-down instructions can be inserted*..." (E.g., see Figure 7 & Column 7, lines 10-21), wherein the potential locations are identified by scanning the code.
- "...*<u>selecting locations to insert the power-down instructions</u> from the <u>identified potential locations</u> in the code based on <u>reducing power consumption</u>*..." (E.g., see Figure 7 & Column 7, lines 39-43, emphasis added), wherein the locations are determined by a predetermined threshold duration of non-use. (emphasis added).

At this point, it should be noted that the Appellant concedes that "Bartley inserts power-down instructions into programming with the goal of reducing power consumption" (*See* brief, page 11, 4th paragraph). Further, as taught above, *Bartley*

selects locations to insert power down instructions from the identified potential locations. Accordingly, Appellants' mere allegation (instant argument) in the context of motivation seems misplaced. In any case, *Bartley* clearly determines where to put the power down instructions to optimize power consumption.

v) Appellant argues (*See* brief, page 12, 2nd, paragraph):

> One relates to hardware design, and the other relates to programming existing hardware. This great difference in architecture and methodology of conserving power makes it highly unlikely that Li et al. would be considered by one of skill in the art when focusing on powering down different functional units. It also places the likelihood of success of such a combination in great jeopardy.

As addressed above in section ii), *Li* expressly discloses "The algorithm is <u>independent of the transformations themselves</u>" (page 4, second paragraph). Additionally, *Li* expressly discloses "Given a set of available transformations techniques, the algorithm needs to: 1. identify *which* transformations can be applied and where, and evaluate these choices of transformations." Here, it is clear that if a transformation or energy optimization can be applied, the user should apply and evaluate it. While, *Li* does focus on adjusting hardware cache design parameters, he does make it clear that the design constraints include software transformations for the purpose of energy optimization (See Li, page 5, indent "2"), wherein the design includes software transformations; and thus *Li* suggests combining his teaching with any available software transformation to optimize energy such as *Bartley's* (i.e. addition of power down instructions to improve energy). Also of note, is the fact that the worst a particular

software transformation could do is not optimize the system energy dissipation and therefore, would not in any way put a combination in "great jeopardy" as concluded above by Appellant.

vi) Appellant argues (*See* brief, page 12, 3rd paragraph) :

"[T]he Final Office Action fails to explain the connection between identifying code segments relating to a functional unit that have a duration longer than a predetermined threshold (which is not user-specified, but rather an inherent aspect of the program code and the associated functional unit), and selecting locations to insert the power-down instructions based on reducing power consumption and satisfying user-specified real-time performance constraints."

Bartley discloses "In the case of either a compiler or assembler, an optimizing process finds, for each functional unit, program segments during which the functional unit is not used are located. The said segments would be of *longer duration* than some *predetermined threshold*, wherein the processor instructions to be inserted are based on the duration of non-activity (longer duration than some predetermined threshold). Herein the predetermined threshold obviously relates to execution time in order to be effective or have any meaningful result. Once these segments are found, the compiler then inserts a power-modifying instruction at the point in the code when the functional unit first goes out of use." (Column 7, lines 42-43). This section of *Bartley*, clearly teaches modifying code depending on time constraints related to execution time in order to reduce power consumption. Although the applied passage does not expressly recite "the number of additional cycles of execution time", the duration of execution time

equals the additional cycles of execution when measured in time. As such, the teaching herein is reasonably interpreted in light of the instant limitation ("*performance of code constraints*") as defined in the specification above, particularly in light of "*other such constraints*" included in the specification definition.

It should also be noted, that a programmer (user) specifying a particular known algorithm to compute the performance constraint threshold, which is then executed by the program is certainly user specified. The fact that the code inherently computes the user specified performance constraint algorithm does not make it not user specified. In fact, Appellant's process is intended to be implemented during compiling time. Accordingly, a programmer (user) specified threshold to be processed by the compiler is typically how a compiler operates and certainly is a reasonable interpretation. If Appellant is attempting to imply that a user interface should solicit input dynamically during compile time and then use the parameter input to compute the algorithm specified by the compiler then Appellant should further define the claim language. However, it is noted that the instant perspective is not supported in the specification.

In any case, *Bartley's* teaching of an execution-time threshold would have been sufficient motivation to one of ordinary skill in the art, at the time the invention was made to consider execution time in relation to instructions to save power as further supported by the arguments addressed in sections ii), iii) and v) above.

vii) Appellant argues (*See* brief, page 13, 1st paragraph) :

"Once again, there is no relation between the determination of a threshold relating to the duration of a program segment for a related functional unit, and the insertion of power-down instructions that satisfy user-specified real-time performance constraints."

In regard to the instant argument, the examiner refers Appellant to the arguments above as addressed in sections ii), iii) and v) accordingly.

viii) Appellant argues (*See* brief, page 13, 2nd paragraph):

"Each threshold appears to be fixed and based on efficiency, not user specified time constraints. Thus, the claim language of inserting power-down instructions while satisfying user-specified real-time constraints does not necessarily flow from the cited language of Bartley, and the rejection should be withdrawn, as at least one element of the claims is lacking from the combination even if proper."

In regard to the instant argument, the examiner refers Appellant to the arguments above as addressed in sections ii), wherein it is noted that *Bartley* is not cited for rejecting the argued claim limitation. Furthermore, see section vi), wherein the fact that the user is the programmer and the fixed code then implements the specified performance constraints computed by the specified algorithm chosen from a set of available transformations (See section v) is proper.

Furthermore, even arguably considering Appellants example embodiment addressed above, on page 12, of the originally filed specification, wherein the user inputs a delta "Δ" of cycles to compute execution time which is then used to determine power down instructions, the examiner notes that this delta used to compute Appellants argument would also be fixed and based on efficiency. Thus, it is unclear how a

specified value, which is then fixed equates to not user specified as argued by Appellant.

In summary, Independent claims 14, 24, and 34 all contain similar recitations and are rejected for at least the same reasons. Accordingly, the rejection of dependent claims 2, 11-15, 22-23, 25, 32-33, 35-36, 43 and are maintained in view of Appellant's instant arguments.

B) Further Arguments Regarding Claims 11, 12, 22, 32 and 43 ***(See brief, pages 14 – 15)***

i) Appellant argues (*See* brief, page 14, 1st paragraph):

> "As pointed out above, Li et al. simply does not deal with power-down instructions at all. Section 5.2 and Table 2 of Li et al. similarly do not mention the use of power down instructions."

Li's teachings of "minimum energy dissipation while not exceeding the budget of clock cycles" (execution time constraint), *See Li*, Section "5.2 Optimizing system-level energy dissipation"; is relied upon to have been obvious to one of ordinary skill in the art to teach the "number of additional cycles of execution time the user is willing to incur" due to a software transformation. Although *Li*, does not expressly state the number of clock cycles, it would have been obvious to one of ordinary skill in the art that a budget of clock cycles allowed, is a number of clock cycles allowed, as taught by *Li* in section 5.2, and illustrated in Table 2. Also, in order to have a "budget" the programmer /user

inherently must specify the budget which again should be noted, is equated by Appellant to equal "execution time constraint".

The claim limitations of "number of power down instructions that can be inserted in an execution path" is determined by *Bartley* as disclosed in claim 1 (See previous rejection or section "iv") above), wherein *Bartley's* disclosure of determining potential locations to insert power down instructions along with subsequently determining where to insert the instructions from the identified potential locations, inherently or necessarily determine the "number of power down instructions that can be inserted in an execution path, including one or more identified potential locations".

ii) Appellant argues (See *brief*, page 14, 2nd paragraph):

"The user-specified number of power-down instructions and additional cycles of execution time recited in these claims relate to the extra execution time of the processor caused by the addition of the power down instructions."

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., "relate to the extra execution time of the processor caused by the addition of the power down instructions") are based on Applicant's interpretation of such instant limitation upon which applicant relies ("performance of code constraints") see response (page 13, second paragraph), Applicant specifically argues that the instant limitation relates to the "extra execution time of the processor caused by the addition of the power down instructions". At this point, it should be noted that this interpretation is not recited in the rejected

claim(s). Again as clarified in previous rejections and above, the plain language of the claims merely recite "performance of code constraints".

iii) Appellant argues (See brief, page 14, 3rd paragraph – Page 15, 1st paragraph):

"A further distinction includes the lack of concern in Bartley for real time performance of the executing code. The claims clearly indicate reduction of power consumption while satisfying real time performance constraints related to execution of the code."

It is noted that Appellant's argument is misleading. As defined in the originally filed disclosure (See specification, page 12, lines 10-13), "The user-specified real-time constraints can include constraints such as the number of power down instructions that can be inserted in an execution path, the number of additional cycles of execution time the user is willing to incur, and other such constraints". Accordingly, Appellant's user specifies a real-time "execution time constraint" and then the constraint is evaluated in view of power consumption when the code is executed. It is important to differentiate this position from the process taking place during execution of the code, rather in both Bartley, Li and Appellants disclosure, the constraint is fixed and then evaluated based on the performance of the executing code in real-time. This is clear form Appellants preamble in claim 1, wherein the "method of compiling" is disclosed, which is clearly not during execution of the code.

In summary, claims 11, 12, 22, 32 and 43 are rejected for at least the above reasons. Accordingly, the are maintained in view of Appellant's instant arguments.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

John Romano

Conferees:

Tuan Dam, SPE 2192

TUAN DAM
SUPERVISORY PATENT EXAMINER

Eddie Lee, SPE 2100

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER